Loops and Recursion

CSE 5211 Analysis of Algorithms

Spring 2017 (January 15, 2017)

Problem Set 2: Due Friday of Week 3

1. The pseudo-code below implements counting sort on an array A of n integers that lie in a range $[0, u)$ for some upper bound u. It counts the number times each value from 0 to $u - 1$ occurs. It then refills the array A with that number of each value.

```
1   void countingSort(array A, int n, int u) {
2     int counts[u];
3     for (int i = 0; i < u; i++) { counts[i] = 0; }
4     for (int i = 0; i < n; i++) { counts[A[i]] = counts[A[i]] + 1; }
5     int index = 0
6     for (int i = 0; i < u; i++) {
7       for (int j = 0; j < counts[i]; j++) {
8         A[index] = i
9         index = index + 1
10      }
11    }
12  }
```

   (a) Trace the algorithm on the array A of length $n = 10$ where $u = 5$.

$$A = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 2 & 4 & 3 & 2 & 4 & 3 & 0 & 1 & 0 & 4 \\ \hline \end{array}$$

   That is, show values in the **counts** array after line 4. And, show the values in the A array each time the loop at line 7 completes.

   (b) What is the run time of the algorithm?

2. The power function computes x^n, where x is a floating point number and n is a natural number.

   (a) What is the run time complexity of this brute-force algorithm?

```
double power(double x, unsigned n) {
  double product = 1.0;
  for (int i = 0; i < n; i++) {
    product = product * x;
  }
  return product;
}
```

(b) Another algorithm computes x^n by taking advantage of the identity

$$x^n = \begin{cases} (x * x)^{n/2} & \text{if } n \text{ is even} \\ x * (x * x)^{\lfloor n/2 \rfloor} & \text{if } n \text{ is odd} \end{cases}$$

What is the worst case time complexity of the algorithm below?

```
power :: Num a => a -> Int -> a
power _ 0 = 1
power x 1 = x
power x n
  | n `rem` 2 == 0 = y
  | otherwise      = x * y
  where y = power (x*x) (n `div` 2)
```